
# Sommario

- Servlet
  - Introduzione alle API ed esempi
- Tomcat
  - Server per applicazioni web

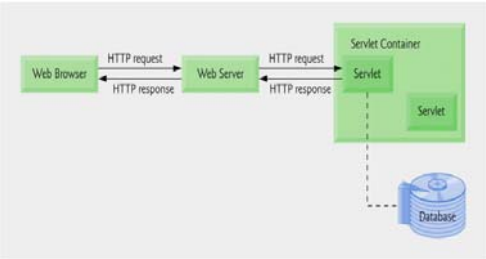
# Java: da applet a servlet

- In origine Java nasce per essere eseguito lato client da un browser (applet) o sul desktop (application)
- Il fatto che Java viene inizialmente compilato in bytecode poi interpretato dalla Java Virtual Machine ci fa pensare di aver finalmente raggiunto il massimo della portabilità
- La portabilità di un componente software ne permette l'uso in un ambiente di esecuzione diverso da quello originale
- WORA - Write Once Run Anywhere
- Ma in realtà … WODE - Write Once Debug Everywhere
- Mentre continua il successo a livello di application le applet sono ormai quasi scomparse dalla rete.
- Java però ha ora preso posizione sul lato server (Servlet e JSP)



# Servlet - Introduzione

- I servlet sono moduli che estendono le funzionalità dei server (es. i server web con supporto Java)
- Un servlet può occuparsi, per esempio, di accettare i dati di un form html e aggiornare un database
- I servlet possono essere utilizzati in diversi tipi di server



# Definizioni

- API
  - Le Application Programming Interface API (Interfaccia di Programmazione di un'Applicazione), sono insiemi di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per un determinato compito.
- Framework
  - Nella produzione del software, il framework è una struttura di supporto su cui un software può essere organizzato e progettato. Alla base di un framework c'è sempre una serie di librerie di codice utilizzabili con uno o più linguaggi di programmazione, spesso corredate da una serie di strumenti di supporto allo sviluppo del software, come ad esempio un IDE, un debugger, o altri strumenti ideati per aumentare la velocità di sviluppo del prodotto finito.

*Wikipedia*

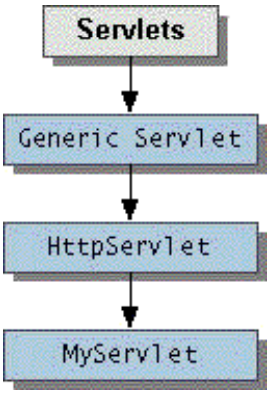


# Scrittura di un servlet

- I servlet sono scritti interamente in Java
- Per svilupparli sono necessarie le due librerie:
  - javax.servlet
  - javax.servlet.http
- Oltre al compilatore java è necessario
  - JDK (per sviluppo di applicazioni Java)
  - JSDK (per sviluppo JSP e servlet)

# Servlet - Servlet API

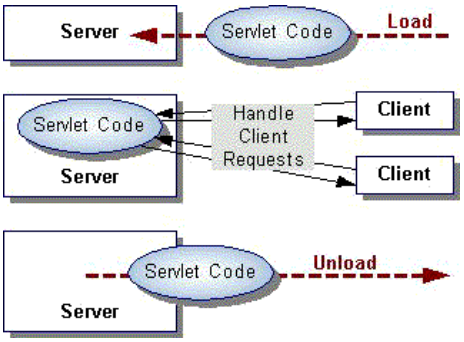
- Il package *javax.servlet* fornisce interfacce e classi per scrivere servlet
- L'astrazione centrale nelle API è l'interfaccia *Servlet*
  - Tutte i servlet implementano questa interfaccia, direttamente oppure estendendo una classe che la implementa, (es. *HttpServlet)*
  - L'interfaccia *Servlet* dichiara (ma non implementa) metodi per gestire servlet e le sue comunicazioni con i client
  - Gli autori di servlet devono implementare questi metodi
- Un servlet deve costruire l'intera pagina dal tag <html> al tag </html>






# Servlet - Ciclo di vita

- Inizializzazione
  - Il server carica e inizializza il servlet
- Esecuzione
  - Quando arriva una richiesta
    - Si riceve un oggetto *ServletRequest* con tutte le informazioni sulla richiesta
    - Si usa un oggetto *ServletResponse* per restituire la risposta
- Distruzione
  - Il server rimuove il servlet



# Servlet
# Gestione richiesta GET

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloUserServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
  HttpServletResponse response) throws IOException,
  ServletException {
      String user = request.getParameter("user");
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<html>");
      out.println("<head><title>Hello Servlet</title></head>");
      out.println("<body><h1>Hello " + user + "!</h1></body>");
      out.println("</html>");
  }
}
```



# Servlet
# Inviare una richiesta GET

```
<form action="http://myhost.com/mycontext/HelloUser"
  method="get">
User: <input type="text" name="user" value="" size="10"
  maxlength="10" /><br />
<input type="submit" name="okbutton" value="OK, submit!" />
<input type="reset" value="Whoops - erase that" />
</form>
```

# Servlet
# Aggiungere parametri ai link

- Quando si clicca su un link in una pagina web, il browser invia una richiesta GET
- Si possono aggiungere alla richiesta dei parametri
  - Coppie nome/valore
  - Devono essere aggiunti alla url della richiesta, dopo un carattere “*?*”
  - Se si passano più parametri, devono essere separati da un “*&*”

```
<a
href="http://myhost.com/mycontext/HelloUser?user=Al&okbutton=OK,+sub
mit!">
```



# Servlet
# Gestire una richiesta POST

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloUserServlet extends HttpServlet {
  public void doPost(HttpServletRequest request,
  HttpServletResponse response) throws IOException,
  ServletException {
      String user = request.getParameter("user");
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<html>");
      out.println("<head><title>Hello Servlet</title></head>");
      out.println("<body><h1>Hello " + user + "!</h1></body>");
      out.println("</html>");
  }
}
```

# Servlet
# Inviare una richiesta POST

```
<form action="http://myhost.com/mycontext/HelloUser"
  method="post">
User: <input type="text" name="user" value="" size="10"
  maxlength="10" /><br />
<input type="submit" name="okbutton" value="OK, submit!" />
<input type="reset" value="Whoops - erase that" />
</form>
```



# Servlet - Gestione sia di GET che di POST

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class PostToGetExample extends HttpServlet {
  public void doGet(HttpServletRequest request,
  HttpServletResponse response) throws IOException,
  ServletException {
      // Handle both GET and POST requests here
  }
  public void doPost(HttpServletRequest request,
  HttpServletResponse response) throws IOException,
  ServletException {
      doGet(request, response);
  }
}
```

# Tomcat



## Tomcat - Introduzione

- Apache Tomcat (o semplicemente Tomcat) è una applicazione open source sviluppata da Apache Software Foundation.
- Implementa le specifiche JSP e Servlet di Sun Microsystems, fornendo quindi una piattaforma per l'esecuzione di applicazioni Web sviluppate nel linguaggio Java. La sua distribuzione standard include anche le funzionalità di web server tradizionale, che corrispondono al prodotto Apache.
- In passato, Tomcat era gestito nel contesto del Jakarta Project, ed era pertanto identificato con il nome di Jakarta Tomcat; attualmente è oggetto di un progetto indipendente.
- Tomcat è rilasciato sotto licenza Apache Software License, ed è scritto interamente in Java; può quindi essere eseguito su qualsiasi architettura su cui sia installata una JVM.

*Wikipedia*

# Tomcat - Riferimenti

- Sito ufficale
  - http://tomcat.apache.org/
- Guida in italiano
  - http://www.mrwebmaster.it/tomcat/guide/guida-tomcat_37/



# Tomcat - Principali cartelle

- */bin* – Vari script per avvio, chiusura ecc. File **.sh* (per sistemi Linux/Unix) e **.bat* (per sistemi Windows)
- */conf* – File di configurazione. *server.xml* per la configurazione del container
- */lib* – Librerie comuni. *servlet-api.jar* deve essere aggiunta al classpath java per compilare le servlet
- */logs* – File di log
- */webapps* – Qui vanno inserite le applicazioni web. Creare una nuova sottocartella per ogni nuova applicazione web

# Tomcat - Organizzazione cartelle

- Bisogna organizzare i file delle applicazioni web come previsto dal formato WAR (Web Application Archive)
- Partiamo dalla cartella principale della nostra applicazione web, contenuta in *webapps*:
  - **.html, *.gif, *.jsp, *.js, *.css, etc.* – Per le applicazioni più semplici, si possono tenere tutti i file visibili al browser nella radice
  - */WEB-INF/* – Tale cartella deve sempre esistere, anche se vuote. Notare il nome in maiuscolo.
  - */WEB-INF/web.xml* – Il descrittore dell'applicazione
  - */WEB-INF/classes/* – Contiene le classi richieste dall'applicazione
    - File *.class e risorse associate (sia servlet che altro) non inclusi in file JAR
    - Come sempre, occorre che le cartelle riflettano l'organizzazione delle classi in package
  - */WEB-INF/lib/* - File JAR necessari, ad esempio librerie di terze parti, driver per database ecc.



# Tomcat - Semplice descrittore (web.xml)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
  http://java.sun.com/xml/ns/j2ee/web-app_2_5.xsd"
    version="2.5">
    <servlet>
        <servlet-name>hello</servlet-name>
        <servlet-class>HelloUserServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>/HelloUser</url-pattern>
    </servlet-mapping>
</web-app>
```